### hdfs

Allows you to browse HDFS or transmit the input records to the file.

#### Syntax

hdfs PROFILE SUBCOMMAND [OPTIONS] PATH

Required Parameter

**PROFILE**

HDFS connect profile. You can configure the profile in the web console.

**SUBCOMMAND**

Command to be executed in the FTP session: ls, cat, put.

* ls: Lists all information about the files in the path specified by PATH.
* lsr: Recursively lists all the files in the directory specified by PATH.
* cat: Loads the contents of text files, CSV files, JSON files, HDFS sequence, and plain text files in the HDFS file system. It parses according to the file format specified by the format option.
* put: Transmits the values of the field specified by the fields option to the HDFS file system.
* rm: Removes the file in the path specified in the input record.

**PATH**

Path to a directory or file. If you use a wildcard (\*) int the file name, you can retrieve all files containing a specific string pattern in the file name(e.g. /var/log/httpd/\*).

* When **SUBCOMMAND** is ls, you can enter either a directory or a file path.
* When **SUBCOMMAND** is cat, you can enter only the file path.
* When **SUBCOMMAND** is put, you can enter only the file path.
* When **SUBCOMMAND** is rm, the PATH is not required.

Optional Parameter

The options for each SUBCOMMAND are as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Options | `cat` | `put` | `ls`/`lsr`/`rm` |
| append | - | O | - |
| compression\_type | - | O | - |
| fields | - | O | - |
| flush | - | O | - |
| format | O | O | - |
| key\_field | - | O | - |
| key\_type | - | O | - |
| limit | O | - | - |
| offset | O | - | - |
| partition | - | O | - |
| value\_field | - | O | - |
| value\_type | - | O | - |

**append=BOOL**

Enables or disables appending data to the end of the file specified in the PATH (default: f).

* t: Appends the field records to the end of the file specified as PATH.
* f: NOT append the field records to the end of the file specified as PATH. The query fails if the file exists.

**compression\_type=TYPE**

Compression type: either block or record (default: no compression).

* block: block-by-block compression
* record: record-by-record compression

**fields=FIELD,...**

Fields to be transmitted to the HDFS server (default: line). Use comma (,) without any leading or trailing whitespaces as a separator. If there is no line field or the specified field is empty, it is replaced with a hyphen symbol (-) in the output to indicate the field is empty.

**flush=INT{y|mon|w|d|h|m|s}**

Cycle to flush output buffer to the file specified as PATH. You can use one of the cycle units of y (year), mon (month), w (week), d (day), h (hour), m (minute), and s (second). For example, to flush the buffer every 5 seconds, specify 5s.

**format=FORMAT**

File format (csv, json, sequence, tsv).

* csv, tsv
* When **SUBCOMMAND** is cat, the first line is considered a regular record. Field name (column header) is assigned in the form columnN (N is a number starting from 0).
* When **SUBCOMMAND** is put, field names (column header) are assigned with the field names specified by the fields option.
* json
* When **SUBCOMMAND** is cat, it parses the file into the records of key-value pairs line by line. Field names are specified as keys and field values as values.
* When **SUBCOMMAND** is put, it transmits the records consisting of the key-value pairs of the fields specified by the fields option. If the fields option is not specified, records consisting of all field values are transmitted.
* sequence
* When **SUBCOMMAND** is cat, it converts the Writable implementation of HDFS to a Logpresso type (the data type of Java) and reads the file record by record.
* The key field is named **key**. The key is converted to a string regardless of its original type.
* When the value field is of type MapWritable, the internal key-value mapping is returned to the field of the returned row. The Hadoop's Writable implementation is converted into a Logpresso type.
* When the value field is not a MapWritable type, it outputs the value to the value field.
* When **SUBCOMMAND** is put, it transmits the file in HDFS sequence format unless it falls under the following conditions:
* When either key or value of the record is empty, that row is not transmitted.
* When the type of value does not match the type specified by the value\_type option, string type is converted to a string, and numeric types such as int, long, float, and double are converted to 0, and boolean type to false.
* When it can convert the type of value without compromising precision, it converts it to the specified type and outputs (for example, when a long type is specified with the value\_type option but an int value comes in, it is converted to a long type and returned).
* Not specified (plain text):
* When **SUBCOMMAND** is cat, values are loaded in the **line** field line by line.
* When **SUBCOMMAND** is put, the file is transmitted in plain text format. Values are separated by tab characters in plain text, and empty values (nulls) are replaced with hyphens (-).

**key\_type=HDFS\_TYPE**

HDFS type in the HDFS data conversion type of Logpresso.

**key\_field=KEY\_FIELD**

Name of the key field. If you do not set this option, the LongWritable counter, which starts from 1, is used.

**limit=INT**

Number of records to be output when importing files (default: unlimited).

**offset=INT**

Number of records to skip when importing files (default: 0)

**partition=BOOL**

Option to enable macro in the PATH (default: f).

* t: Enables macro
* f: Disables macro

You can specify PATH to change the directory and file path over time using a macro when partition=t. The available macros are {logtime:FMT} and {now:FMT}.

* {logtime:FMT}: Names the directory or file based on the log occurrence time.
* {now:FMT}: Names the directory or file based on the current time.

If you set 'partition=t' and do not use a macro on the path, the query fails.

**value\_type=HDFS\_TYPE**

HDFS type in the HDFS data conversion type of Logpresso.

**value\_field=VALUE\_FIELD**

Name of the value field. If you do not set the name of the value field, all fields are transmitted to a single MapWritable.

#### Description

Logpresso uses the data types defined by Logpresso, such as Java standard data types and IP addresses. When importing or transmitting data from HDFS, Logpresso performs the conversion operation according to the HDFS data type. For information on converting data by type, refer to the following table.

**Logpresso and HDFS data conversion type**

|  |  |  |
| --- | --- | --- |
| Logpresso type | HDFS type | Description |
| String | Text | String |
| Null | NullWritable | Null |
| Boolean | BooleanWritable | Boolean |
| Integer | IntWritable, VIntWritable | 4-byte (32 bits) integer |
| Long | LongWritable, VLongWritable | 8-byte (64 bits) integer |
| Float | FloatWritable | Single precision real number |
| Double | DoubleWritable | Double precision real number |

#### Usage

Retrieve the root path file list by accessing the profile of the name vm.

hdfs vm ls /

The output fields are as follows:

* **type** (string): "dir" when it is a directory, "file" when it is a file
* **name** (string): File name
* **path** (string): Absolute path of the file
* **replication** (integer): Number of copies, 0 when it is a directory
* **file\_size** (integer): File size, 0 when it is a directory
* **block\_size** (integer): Block size, 0 when it is a directory
* **modified\_at** (date): Last modified time
* **permission** (string): Permission settings
* **owner** (string): Owner
* **group** (string): Owned group

Read 5 rows after skipping the first line of the /tmp/LICENSE.txt file by accessing the vm profile.

hdfs vm cat offset=1 limit=5 /tmp/LICENSE.txt

Read 3 rows of the /tmp/malware.csv file by accessing the vm profile.

hdfs vm cat format=csv limit=3 /tmp/malware.csv

Read 1 row of the /tmp/iis.json file by accessing the vm profile.

hdfs vm cat format=json limit=1 /tmp/iis.json

Read 2 records of the /tmp/classloading.seq file by accessing the vm profile.

hdfs vm cat format=sequence limit=2 /tmp/classloading.seq

Output only UnloadedClassCount of LoadedClassCount among the JMX class loading logs to the /tmp/class.txt path.

table classloading | hdfs vm put fields=UnloadedClassCount,LoadedClassCount /tmp/class.txt

Output the sys\_cpu\_logs log to the directory under /tmp by date.

table sys\_cpu\_logs | eval line=concat("idle: ", idle, ", kernel: ", kernel, ", user: ", user) | hdfs vm put partition=t /tmp/{logtime:yyyyMMdd}/cpu.txt

Output LoadedClassCount, UnloadedClassCount, and TotalLoadedClassCount among the JMX class loading logs.

table classloading | hdfs vm put format=csv fields=LoadedClassCount,UnloadedClassCount,TotalLoadedClassCount /tmp/classloading.csv

Output the JMX class loading log as a JSON file.

table classloading | hdfs vm put format=json /tmp/classloading.json

Output the entire JMX class loading log as an HDFS sequence file.

table classloading | hdfs vm put format=sequence /tmp/classloading.seq

Output LoadedClassCount among the JMX class loading logs.

table classloading | hdfs vm put format=sequence value\_type=long value\_field=LoadedClassCount /tmp/classloading.seq